
Article:
Malvetti, Emanuel, Iten, Raban and Colbeck, Roger orcid.org/0000-0003-3591-0576 (2021) Quantum Circuits for Sparse Isometries. Quantum. 412. ISSN 2521-327X

https://doi.org/10.22331/q-2021-03-15-412

Quantum Circuits for Sparse Isometries

Emanuel Malvetti[1], Raban Iten[2], and Roger Colbeck[3]

[1]Department of Chemistry, Technische Universität München, Lichtenbergstraße 4, 85747 Garching, Germany

[2]ETH Zürich, 8093 Zürich, Switzerland

[3]Department of Mathematics, University of York, YO10 5DD, UK

9th March 2021

We consider the task of breaking down a quantum computation given as an isometry into C-NOTs and single-qubit gates, while keeping the number of C-NOT gates small. Although several decompositions are known for general isometries, here we focus on a method based on Householder reflections that adapts well in the case of sparse isometries. We show how to use this method to decompose an arbitrary isometry before illustrating that the method can lead to significant improvements in the case of sparse isometries. We also discuss the classical complexity of this method and illustrate its effectiveness in the case of sparse state preparation by applying it to randomly chosen sparse states.

1 Introduction

A general quantum computation on an isolated system can be represented by a unitary matrix. In order to execute such a computation on a quantum computer, it is common to decompose the unitary into a quantum circuit, i.e., a sequence of quantum gates that can be physically implemented on a given architecture. There are different universal gate sets for quantum computation. Here we choose the universal gate set consisting of C-NOT and single-qubit gates [1]. We measure the cost of a circuit by the number of C-NOT gates since in many architectures they are more difficult to implement than single-qubit gates. In addition, the number of single-qubit gates is bounded by about twice the number of C-NOTs [2, 3], so the C-NOT counts are illustrative of the total gate counts for this gate set.

It can also be useful to consider operations where the dimensions of the input are different to those of the output. An isometry from m qubits to n qubits can be represented by a $2^n \times 2^m$ matrix V satisfying $V^\dagger V = I$. Unitaries and state preparation are special cases of isometries where $m = n$ or $m = 0$ respectively. An isometry with $m \neq n$ can be implemented by extending it to a unitary and implementing the unitary instead. The freedom in the extension can be exploited to lower the number of gates required. The main aim of the present paper is to consider decompositions that adapt well to the case of sparse isometries, i.e., those with many zero entries in the computational basis.

We briefly summarize previous work on decompositions using the gate set consisting of C-NOT and single-qubit gates. Arbitrary unitaries can be decomposed using $\frac{23}{48}4^n$ C-NOTs [4] to leading order, about twice as many as the best known lower bound. The most efficient known method for preparing arbitrary states requires about 2^n C-NOTs to leading order [5–7], which is about twice the best known lower bound [5]. The decomposition of arbitrary isometries has been considered in [7, 8]. Near optimal methods for decomposing arbitrary isometries exist, and again they achieve C-NOT counts approximately twice as large as the best known lower bounds. The implementation of quantum channels has been considered in [9] and these have been implemented along with POVMs and instruments in [10].

Particular classes of sparse unitaries have been studied in previous work, e.g., diagonal gates [4], uniformly controlled single-qubit gates [6] and permutation gates [11]. The case of efficiently computable sparse unitaries with a polynomial number of non-zero entries per row or column has also been considered in Ref. [12]. The authors showed that these can be implemented using a polynomial number of gates, although explicit decompositions were not given.

In this work we present methods for decomposing isometries based on Householder decompositions, whose significance for (dense) circuit decompositions has been studied previously. Ref. [13] showed that n qubit unitaries to within approximation ε can be decomposed using the Clifford+T gate library with $\mathcal{O}(4^n n(\log(1/\varepsilon)+n))$ gates. Ref. [14] gave a decomposition of unitaries with a speed advantage over methods based on the Cosine-Sine decomposition [4], although using 4^n C-NOTs asymptotically, rather than $\frac{23}{48}4^n$.

For dense isometries our method achieves C-NOT counts that are close to those of [7, 8] (which are near optimal). The advantage of using Householder

Emanuel Malvetti: emanuel.malvetti@tum.de
Raban Iten: itenr@itp.phys.ethz.ch
Roger Colbeck: roger.colbeck@york.ac.uk

decompositions becomes apparent when applied to sparse isometries, leading to C-NOT counts that depend on the number of non-zero entries and the positions of the non-zero elements. Our Householder-based algorithm proceeds one column at a time, replacing each successive column by a computational basis vector. For sparse isometries, the improvement we get over other such decompositions is due to the relatively small amount of fill-in that occurs in each step. By contrast, using the decompositions of [7, 8], which also work one column at a time, the operations performed to replace the first column typically remove all of the sparseness elsewhere.

In the case of state preparation of an n qubit state $|v\rangle$ our decompositions require $\mathcal{O}(n\,\mathrm{nnz}(v))$ C-NOTs, where $\mathrm{nnz}(v)$ is the number of non-zero entries in the state. The counts for isometries beyond state peparation are more complicated because fill-in occurs as the algorithm proceeds, and the amount of fill-in depends on the positions of the non-zero entries. We have three different methods for a sparse isometry W from m qubits to n qubits, the simplest to count requires $\mathcal{O}(n\,\mathrm{nnz}(W))$ C-NOTs. Table 1 summarizes our main results and gives more explicit counts. The main decompositions presented in this work have also been implemented using UNIVERSALQCOMPILER [10].

The remainder of this paper is organized as follows. In Section 2 we discuss the case of state preparation, which is a building-block for the later cases. The main idea behind the decomposition there is to use pivoting gates, which permute the entries of the state such that the non-zero entries are grouped together, in effect reducing to state preparation on a smaller system. In Section 3 we show that Householder reflections with respect to a hyperplane orthogonal to a sparse state $|v\rangle$ can be implemented using sparse state preparation, again using at most $\mathcal{O}(n\,\mathrm{nnz}(v))$ C-NOTs. In Section 4 we show how a general isometry V from m to n qubits can be implemented using Householder reflections before moving to the sparse case, illustrating the advantage of Householder reflections for the latter. In Section 5 we show how the main decompositions of this paper can be implemented and derive the classical complexities of the implementations. Finally, in Section 6 we present C-NOT counts for sparse state preparation found by explicit use of our method, indicating the improvements compared to dense state preparation methods.

2 Sparse State Preparation

State preparation is the main building-block of the decompositions used in this work. In this section we introduce a method for implementing sparse state preparation more efficiently than is possible in the dense case. The main results from this section are summarized in Table 2

Definition 1. Let $|v\rangle$ be a state on n qubits. We say that a unitary SP_v on n qubits implements state preparation for $|v\rangle$ if

$$\mathrm{SP}_v\,|0\rangle_n = |v\rangle\,.$$

We start by presenting a useful pivoting algorithm for permuting entries in a sparse state such that all non-zero entries are grouped together. The idea is to then perform a decomposition scheme for dense state preparation on the grouped entries, which correspond to the state of a subset of the n qubits.

Lemma 2. *Let $|v\rangle$ be a state on n qubits and let $\mathrm{nnz}(v)$ denote the number of non-zero entries of $|v\rangle$ in the computational basis. Let $s = \lceil \log_2 \mathrm{nnz}(v) \rceil$. Then there exists a permutation gate Piv_v that disentangles $n-s$ qubits in a computational basis state, i.e., $\mathrm{Piv}_v\,|v\rangle = |i\rangle_{n-s} \otimes |\tilde{v}\rangle_s$ for some s-qubit state $|\tilde{v}\rangle_s$, and $i \in \{0, 1, \ldots, 2^{n-s}-1\}$.*

Let $\mathcal{N}_{\mathrm{Piv}}(n,s)$ denote the number of C-NOTs required for pivoting any state on n qubits with at most 2^s non-zero entries and let $\mathcal{N}_{C_s^n(X)}$ denote the number of C-NOTs required to implement an s-controlled NOT when there are a total of n qubits. Then

$$\mathcal{N}_{\mathrm{Piv}}(n,s) \leq (n-1+\mathcal{N}_{C_s^n(X)})\,\mathrm{nnz}(v).$$

Explicit counts are given in Table 2.

Proof. If $s = n$ there is nothing to do so assume $s < n$. The given state can be represented as a column vector in the computational basis, where the basis states can be written in terms of n binary indices $|b_1 b_2 \ldots b_n\rangle$, which we split into two $|b_1 \ldots b_{n-s}\rangle\,|b_{n-s+1} \ldots b_n\rangle$. The first of these corresponds to a block of the vector. The goal is then to move all non-zero elements of $|v\rangle$ into a single block. We achieve this by using the following algorithm.

Algorithm 1.

1. If all non-zero entries are in the target block, we stop the algorithm.
2. Pick a non-zero entry outside the target block and a zero entry inside the target block. Write the basis state of the non-zero entry as $|\mathbf{t}'\rangle_{n-s}\,|\mathbf{r}'\rangle_s$ and that of the zero entry as $|\mathbf{t}\rangle_{n-s}\,|\mathbf{r}\rangle_s$.
3. Choose a qubit on which $\mathbf{t}'_{n-s}$ and $\mathbf{t}_{n-s}$ differ.
4. With this as a control qubit, use at most $n-1$ C-NOTs to adjust $|\mathbf{t}'\rangle_{n-s}\,|\mathbf{r}'\rangle_s$ to $|\mathbf{t}''\rangle_{n-s}\,|\mathbf{r}\rangle_s$ such that $\mathbf{t}''$ and $\mathbf{t}$ differ only on the control qubit.
5. Use one s-controlled NOT (controlling on $|\mathbf{r}\rangle_s$) to exchange $|\mathbf{t}''\rangle_{n-s}\,|\mathbf{r}\rangle_s$ and $|\mathbf{t}\rangle_{n-s}\,|\mathbf{r}\rangle_s$. Note that none of the other entries of the target block are affected by this process.
6. Return to Step 1.

Gate (method)	Ancillas	C-NOT count	Reference
State preparation	0 (1 dirty if $s = n-1$)	$(n + 16s - 9)\,\mathrm{nnz}(v) + \frac{23}{24}2^s$	Corollary 5
State preparation	$\lceil \frac{s}{2} - 1 \rceil$ clean	$(n + 6s - 7)\,\mathrm{nnz}(v) + \frac{23}{24}2^s$	Corollary 5
Isometry (basic)	1 dirty	$(17n - 5)\,\mathrm{elim}(W, \rho, \sigma) + (51n + 34m - 44)2^m$	Remark 19
Isometry (basic)	$\lceil \frac{n-3}{2} \rceil$ clean	$(7n - 3)\,\mathrm{elim}(W, \rho, \sigma) + (21n + 24m - 38)2^m$	Lemma 17
Isometry (fixed envelope)	1 dirty	$4\,\mathrm{ed}(\Pi_\rho W \Pi_\sigma) + \mathcal{O}(n2^n)$	Remark 26
Isometry (no fill-in)	1 clean + 1 dirty	$(17n + 12)\,\mathrm{nnz}(W) + (34n + 34m - 5)2^m$	Lemma 27
Isometry (no fill-in)	$\lceil \frac{n}{2} \rceil$ clean	$(7n + 4)\,\mathrm{nnz}(W) + (14n + 24m - 21)2^m$	Lemma 27

Table 1: **Main results of this work.** This table gives C-NOT counts (upper bounds) for the decompositions for a sparse state of n qubits, v, or sparse isometry from m to n qubits, W. For sparse isometries we have three methods (basic, fixed envelope or no fill-in). Here $\mathrm{nnz}(\cdot)$ is the number of non-zero entries of a state or of an isometry in the computational basis and $s = \lceil \log_2 \mathrm{nnz}(v) \rceil$. The number of eliminations $\mathrm{elim}(W, \rho, \sigma)$ when using the elimination order given by the row and column permutations ρ and σ is defined in Eq. (3) and depends on the positions of the non-zero entries in the isometry, which affects how many zero elements become non-zero as the algorithm proceeds. The quantity $\mathrm{ed}(W)$ is defined in Definition 22, and counts the number of elements (zero and non-zero) between the matrix envelope (as defined in Definition 21) and the diagonal, and it also depends on the positions of the non-zero entries in the isometry. By Π_ρ and Π_σ we denote the permutation matrices corresponding to the permutations ρ and σ. By Lemma 23 we have $\mathrm{elim}(W, \rho, \sigma) \leq \mathrm{ed}(\Pi_\rho W \Pi_\sigma)$, and it trivially holds that $\mathrm{ed}(\Pi_\rho W \Pi_\sigma) \leq 2^{m+n}$, but our decompositions are most useful when $\mathrm{elim}(W, \rho, \sigma) \ll 2^{m+n}$ or $\mathrm{ed}(\Pi_\rho W \Pi_\sigma) \ll 2^{m+n}$. An ancillary qubit is called clean if it starts in a known computational basis state and is restored to that state after the computation. It is called dirty if it starts in an unknown state and is restored after the computation.

Gate	Ancillas	Notation	C-NOT count	Reference
Pivoting	0 (1 dirty if $s = n-1$)	Piv_v	$(n + 16s - 9)\,\mathrm{nnz}(v)$	Lemma 2
Pivoting	$\lceil \frac{s}{2} - 1 \rceil$ clean	Piv_v	$(n + 6s - 7)\,\mathrm{nnz}(v)$	Lemma 2
Permuted diagonal isometry	0 (1 dirty if $m = n-1$)	$\Pi_n I_{n,m} \Delta_m$	$(n + 34m - 34)2^m$	Lemma 15[a]
Permuted diagonal isometry	$\lceil \frac{m}{2} - 1 \rceil$ clean	$\Pi_n I_{n,m} \Delta_m$	$(n + 24m - 32)2^m$	Lemma 15
Householder reflection up to $\Delta\Pi$	1 dirty	$\Delta\Pi H_v$	$(n + 16s - 5)\,\mathrm{nnz}(v) + 16n$	Lemma 16
Householder reflection up to $\Delta\Pi$	$\lceil \frac{n-3}{2} \rceil$ clean	$\Delta\Pi H_v$	$(n + 6s - 3)\,\mathrm{nnz}(v) + 6n$	Lemma 16

Table 2: C-NOT **counts (upper bounds) for the additional decompositions introduced in this work.** Here n is the number of output qubits, m the number of input qubits, $\mathrm{nnz}(\cdot)$ the number of non-zero entries of a state or of an isometry in the computational basis and $s = \lceil \log_2 \mathrm{nnz}(v) \rceil$. Furthermore Π_n denotes an arbitrary permutation gate on n qubits, Δ_m denotes an arbitrary diagonal gate on m qubits and $I_{n,m}$ denotes the 2^m first columns of the identity gate on n qubits. All results follow from the given reference and the results in Table 3 and other entries in the present table. Slightly different results can be derived by using different decompositions for multi-controlled NOT gates. An ancillary qubit is called clean if it starts in a known computational basis state and is restored to that state after the computation. It is called dirty if it starts in an unknown state which must be restored after the computation.

[a] Tighter bounds follow by using the more precise counts for permutations in Appendix D.

At the end of this algorithm, all non-zero entries of $|v\rangle$ are in the target block. Thus we used at most $n-1$ C-NOTs and one s-controlled NOT to insert one non-zero entry into the target block. Since no non-zero entry ever leaves the target block, the claimed bound follows. □

Remark 3. In the preceding Lemma we split the vector into blocks, where we have chosen to separate the $n-s$ most significant qubits from the s remaining ones. However, any splitting into $n-s$ and s qubits would work. In addition, the target block and the order in which to proceed are not fixed and none of these choices affect any of the decompositions used in this work. Making these choices in the right way can reduce the C-NOT counts. See Section 5.2 for details on implementing the algorithm.

Remark 4. Using Table 3 the following bounds on the C-NOT count hold if a dirty ancilla are available.

$$\mathcal{N}_{\mathrm{Piv}}(n,s) \leq (n+16s^2-28s-3)\,\mathrm{nnz}(v) \quad \text{for } n+a \geq s+1$$
$$\mathcal{N}_{\mathrm{Piv}}(n,s) \leq 27n2^n \text{ for } n+a \geq s+1$$
$$\mathcal{N}_{\mathrm{Piv}}(n,s) \leq (n+16s-9)\,\mathrm{nnz}(v) \text{ for } n+a \geq s+2$$
$$\mathcal{N}_{\mathrm{Piv}}(n,s) \leq (n+8s-13)\,\mathrm{nnz}(v) \quad \text{for } n+a \geq s+\left\lceil \frac{s}{2} \right\rceil, s \geq 5$$

In the case $n = s+1$, $a = 0$, there are not enough qubits available for any of the known decompositions of an s-controlled NOT and hence we can decompose it as an s-controlled single-qubit gate (first inequality) or note that the entire pivoting can be written as a permutation (second inequality – see Appendix D for a more precise bound). The first of these gives a better count for small n.

This pivoting algorithm allows us to find a scheme with low cost for the preparation of sparse states.

Corollary 5. *Let $|v\rangle$ be a state on n qubits and let $\mathrm{nnz}(v)$ denote the number of non-zero entries of $|v\rangle$ in the computational basis. Let $s = \lceil \log_2 \mathrm{nnz}(v) \rceil$. The number of* C-NOT*s required for sparse state preparation of a state of n qubits with $\mathrm{nnz}(v)$ non-zero elements is bounded by*

$$\mathcal{N}_{\mathrm{SSP}}(n,s) \leq \mathcal{N}^{\Delta}_{\mathrm{Piv}}(n,s) + \mathcal{N}_{\mathrm{SP}}(s),$$

where $\mathcal{N}^{\Delta}_{\mathrm{Piv}}(n,s)$ denotes the number of C-NOT *gates used to implement pivoting up to a diagonal gate, i.e., to implement $\Delta\,\mathrm{Piv}$ for some diagonal gate Δ. Explicit counts are given in Table 2.*

Proof. It is sufficient to find a circuit that maps $|v\rangle$ to $|0\rangle_n$, since the inverse of this circuit implements state preparation for $|v\rangle$. Let Piv_v be constructed as in Lemma 2, then $\Delta\,\mathrm{Piv}_v\,|v\rangle$ has the form $|i\rangle_{n-s} \otimes |\tilde{v}\rangle$ for any diagonal gate Δ. Without loss of generality $i = 0$. Now we merely need to apply reverse state preparation for $|\tilde{v}\rangle$ on s qubits. Thus sparse state preparation can be implemented as

$$\mathrm{SSP}_v = (\Delta\,\mathrm{Piv}_v)^{\dagger}(I_{n-s} \otimes \mathrm{SP}_{\tilde{v}}), \tag{1}$$

which gives the claimed count. □

Remark 6. For use in our sparse state preparation decomposition, it is sufficient to decompose the s-controlled NOT gates of Lemma 2 up to a diagonal gate. For example the Toffoli gate (with two controls) requires six C-NOTs when implemented exactly [1], but it can be implemented using only three C-NOTs when implemented up to a diagonal gate [1, Section VI B]. We are not currently aware of schemes to decompose NOT gates with more controls up to diagonal, but, if these were found, our counts would be improved.

Remark 7. A more straightforward way to implement sparse state preparation is to use two-level unitaries to eliminate the non-zero entries one by one, but this leads to higher counts than the method presented here. A similar method is used in [1] to implement arbitrary unitaries.

3 Generalized Householder Reflections

Given a unit vector $|v\rangle$, the standard Householder reflection [16] with respect to $|v\rangle$ is defined as

$$H_v = I - 2\,|v\rangle\langle v|\ .$$

We call $|v\rangle$ the Householder vector associated with the reflection. The generalized Householder reflection of phase ϕ with respect to $|v\rangle$ is defined as

$$H_v^{\phi} = I + (\mathrm{e}^{\mathrm{i}\phi} - 1)\,|v\rangle\langle v|$$

and coincides with the standard definition if $\phi = \pi$. On certain architectures generalized Householder reflections can be implemented directly [17] and in a fault-tolerant way [18]. Standard Householder reflections can be approximated well using Clifford and T gates [13]. In the circuit model a state preparation scheme can be used to perform a generalized Householder reflection.

Lemma 8. *Let SP_v denote a unitary implementing state preparation for the state $|v\rangle$ and H_0^{ϕ} the Householder reflection with respect to $|0\rangle$. Then $H_v^{\phi} = \mathrm{SP}_v \cdot H_0^{\phi} \cdot \mathrm{SP}_v^{\dagger}$.*

Proof. This can be seen by considering the action on an orthonormal basis containing $|v\rangle$. □

Lemma 9. *The gate H_0^{ϕ} on n qubits can be implemented using an $(n-1)$-controlled single-qubit gate. The special case H_0 can be implemented using the same number of* C-NOT*s and ancilla qubits as an $(n-1)$-controlled* NOT *gate. Explicit counts for these gates are given in Table 3.*

Gate	Ancillas	Notation	C-NOT count	Reference
Diagonal gate	0	Δ_n	$2^n - 2$	[4]
State preparation	0	SP_v	$\frac{23}{24}2^n - 2^{\frac{n}{2}+1} + 5/3$	[5], [7, Remark 5][a]
k-controlled single-qubit gate	0	$C_k(U)$	$16k^2 - 28k - 2$ if $k \geq 2$	[7, Theorem 4]
k-controlled single-qubit gate	$k-1$ clean	$C_k(U)$	$6k - 4$	[15, Corollary 4]
k-controlled NOT gate	1 dirty	$C_k(X)$	$16k - 8$	Corollary 30, [7]
k-controlled NOT gate	$\lceil \frac{k}{2} - 1 \rceil$ dirty	$C_k(X)$	$8k - 12$ if $k \geq 5$	[15, Proposition 5]
k-controlled NOT gate	$\lceil \frac{k}{2} - 1 \rceil$ clean	$C_k(X)$	$6k - 6$ if $k \geq 2$	[15, Proposition 4]
Permutation gate	0	Π_n	$27n2^n$	Appendix D, [11][b]
Permutation gate	1 dirty	Π_n	$(18n - 26)2^n$	Appendix D[b]

Table 3: **Useful C-NOT counts (upper bounds) for some basic gates.** In each case n denotes the number of qubits on which the gate acts and k denotes the number of control qubits. An ancillary qubit is called clean if it starts in a known computational basis state and is restored to that state after the computation. It is called dirty if it starts in an unknown state which must be restored after the computation.

[a] For convenience of presentation we have consolidated the bounds for even and odd n into a single bound.

[b] See Appendix D for tighter bounds.

Proof. The gate H_0^ϕ is a multi-controlled single-qubit gate with $n-1$ controls and hence the C-NOT count is as in Table 3. If $\phi = \pi$, then using some common gates

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

(note the Hadamard gate H has a similar notation to the Householder reflection) and

$$-Z = XZX = X(HXH)X$$

we obtain

$$H_0 = C_{n-1}(-Z) = XH \cdot C_{n-1}(X) \cdot HX\,.$$

The C-NOT count is thus the one of $C_{n-1}(X)$ (see again Table 3). □

Given two states $|v\rangle$ and $|w\rangle$ we can construct a gate that maps $|v\rangle$ to $\mathrm{e}^{\mathrm{i}\theta}|w\rangle$ for some real θ using a standard Householder reflection defined as

$$H_{v,w} = H_u, \quad \text{where } |u\rangle = \frac{|v\rangle - \mathrm{e}^{\mathrm{i}\theta}|w\rangle}{\| |v\rangle - \mathrm{e}^{\mathrm{i}\theta}|w\rangle \|} \tag{2}$$

with $\theta = \pi - \arg(\langle v|w\rangle)$ or $\theta = 0$ if $\langle v|w\rangle = 0$. We also define the generalized Householder reflection

$$\tilde{H}_{v,w} = H_u^\phi, \quad |u\rangle = \frac{|v\rangle - |w\rangle}{\| |v\rangle - |w\rangle \|}, \quad \mathrm{e}^{\mathrm{i}\phi} = \frac{\langle v|w\rangle - 1}{1 - \overline{\langle v|w\rangle}},$$

which has the property

$$\tilde{H}_{v,w}|v\rangle = |w\rangle\,.$$

The motivation for these definitions and related proofs are given in Appendix A.

4 Householder Decomposition

Our goal is to find a decomposition of any isometry V from m to n qubits. Let $I_{n,m}$ denote the first 2^m columns of the identity gate on n qubits. If $G = G_k \cdots G_1 G_0$ is a product of elementary gates on n qubits such that $GV = I_{n,m}$, then $G^\dagger$ is an extension of V to a unitary. Thus $G^\dagger$ yields an implementation of V using elementary gates.

Householder reflections provide a straightforward method for implementing arbitrary isometries. Let $|v_0\rangle = V|0\rangle$ be the first column of V and consider $H_{v_0,0}$, the Householder reflection mapping $|v_0\rangle$ to $|0\rangle$ up to a phase. We can reduce the first column (and row by orthogonality) of V by applying the Householder reflection to the isometry, i.e., the only entry in the first row and column of $H_{v_0,0}V$ is that corresponding to $|0\rangle\langle 0|$. Using the same idea the isometry can be reduced column by column to a diagonal isometry. Applying a diagonal gate on m qubits then yields $I_{n,m}$. See Figure 1 for a schematic representation of the decomposition.

Before we describe the decomposition in more detail we show how the reduction of a column via Householder reflection affects the other columns of the isometry.

Lemma 10. *Let V be an isometry. Let $|v_j\rangle = V|j\rangle$ be the j^{th} column of V and i be the target row index. The Householder reflection $H_{v_j,i}$ reduces the j^{th} column to the i^{th} row (i.e., such that its only non-zero element is in the i^{th} row). For $s \neq i$ and $t \neq j$ we have*

$$\langle i|H_{v_j,i}V|t\rangle = \langle s|H_{v_j,i}V|j\rangle = 0, \quad \langle i|H_{v_j,i}V|j\rangle = \mathrm{e}^{\mathrm{i}\theta}$$

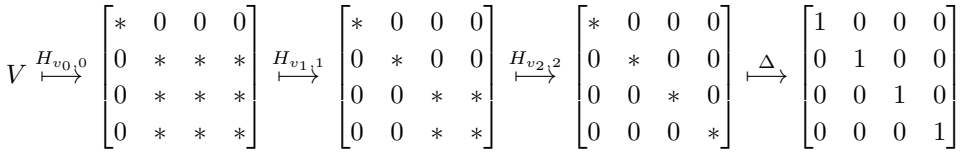

Figure 1: **Basic idea of the Householder decomposition for dense isometries.** Here $*$ represents an arbitrary complex entry. Each step reduces one column without affecting the previous columns. The rows are reduced automatically due to the orthogonality of the columns. The final diagonal gate sets the phases on the diagonal equal to one.

and

$$\langle s|\, H_{v_j,i} V\, |t\rangle = \langle s|V|t\rangle + \mathrm{e}^{-\mathrm{i}\theta}\, \frac{\langle s|V|j\rangle\, \langle i|V|t\rangle}{1 + |\, \langle i|V|j\rangle\,|}$$

where $\theta = \pi + \arg(\langle i|V|j\rangle)$ *or* $\theta = 0$ *if* $\langle i|V|j\rangle = 0$.

Proof. By construction we have $H_{v_j,i} V\, |j\rangle = \mathrm{e}^{\mathrm{i}\theta}\, |i\rangle$ and by orthogonality of the columns $\langle i|\, H_{v_j,i} V\, |t\rangle = 0$. For the final case given in the statement, we compute

$$\begin{aligned}
&\langle s|H_{v_j,i} V\, |t\rangle \\
&\quad = \langle s|\left(I - 2\frac{(|v_j\rangle - \mathrm{e}^{\mathrm{i}\theta}\, |i\rangle)(\langle v_j| - \mathrm{e}^{-\mathrm{i}\theta}\, \langle i|)}{2(1 + |\, \langle v_j|i\rangle\,|)}\right) V\, |t\rangle \\
&\quad = \langle s|V|t\rangle - \langle s|\, \frac{|v_j\rangle\, (-\mathrm{e}^{-\mathrm{i}\theta})\, \langle i|}{1 + |\, \langle i|V|j\rangle\,|} V\, |t\rangle \\
&\quad = \langle s|V|t\rangle + \mathrm{e}^{-\mathrm{i}\theta}\, \frac{\langle s|V|j\rangle\, \langle i|V|t\rangle}{1 + |\, \langle i|V|j\rangle\,|}.
\end{aligned}$$

□

Corollary 11. *Let* $s \neq i$ *and* $t \neq j$, *then* $\langle s|\, H_{v_j,i} V\, |t\rangle \neq \langle s|V|t\rangle$ *if and only if* $\langle i|V|t\rangle$ *and* $\langle s|V|j\rangle$ *are non-zero.*

In other words if the j^{th} column is reduced to the i^{th} row then the entry of V at position s, t does not change if $\langle i|V|t\rangle = 0$ or $\langle s|V|j\rangle = 0$.

4.1 Dense isometries

First we consider the Householder decomposition for dense isometries. This decomposition works for any isometry, but does not take advantage of any sparseness. The idea is to reduce the columns one by one. It follows from Corollary 11 that previously reduced columns are not affected by subsequent Householder reflections. More precisely define $V_0 = V$ and iteratively

$$|v_i\rangle = V_i\, |i\rangle\,, \quad G_i = H_{v_i,i}, \quad V_{i+1} = G_i V_i,$$

for $i = 0, \ldots, 2^m - 1$. Then $G_{2^m-1} \cdots G_0 V = I_{n,m} \Delta_m$ where Δ_m denotes a diagonal gate on m qubits. This proves the following lemma.

Lemma 12. *Let* V *be an isometry from* m *to* n *qubits. Then* V *can be implemented using* 2^m *Householder reflections on* n *qubits and a diagonal gate on* m *qubits.*

In any sequence of Householder reflections implemented using the construction in Lemma 8, gates of the form $\mathrm{SP}_v^\dagger \cdot \mathrm{SP}_w$ occur. Using the dense state preparation scheme from [5] one can merge some gates as described in [7, Theorem 1] for Knill's decomposition [8]. This essentially halves the C-NOT count. The structural similarity between the Householder decomposition and Knill's decomposition suggests a generalized decomposition, which we present in Appendix E. The proofs of the following two results are given in Appendix B.

Lemma 13. *Let* V *be an isometry from* m *to* n *qubits with* $n \geq 5$. *Then* V *can be implemented via standard Householder reflections using one dirty ancilla qubit and* $\mathcal{N}_{\text{iso}}(m, n)$ C-NOT *gates where*

$$\begin{aligned}
\mathcal{N}_{\text{iso}}(m,n) &\leq \tfrac{23}{24}(2^{m+n} + 2^n) + \tfrac{23}{12} 2^{m+\frac{n}{2}} - 2^{m+\frac{n}{4}+2} \\
&\quad + (16n - 23)2^m - 2/3 \quad \textit{if } n \textit{ is even}, \\
\mathcal{N}_{\text{iso}}(m,n) &\leq \tfrac{115}{96}(2^{m+n} + 2^n) + \tfrac{23}{12} 2^{m+\frac{n-1}{2}} - 2^{m+\frac{n-1}{4}+2} \\
&\quad + (16n - 23)2^m - 2/3 \quad \textit{if } n \textit{ is odd}.
\end{aligned}$$

Note that, to leading order, the counts match those of [7, Theorem 1] and hence, in the case of dense isometries, the advantage of using the Householder decomposition rather than Knill's is only apparent for small cases.

Corollary 14. *Let* U *be a unitary on* n *qubits with* $n \geq 5$. *Then* U *can be implemented via standard Householder reflections using one dirty ancilla qubit and* $\mathcal{N}_U(n)$ C-NOT *gates where*

$$\begin{aligned}
\mathcal{N}_U(n) &\leq \tfrac{161}{240} 4^n + \mathcal{O}\left(2^{3n/2}\right) \quad \textit{if } n \textit{ is even}, \\
\mathcal{N}_U(n) &\leq \tfrac{23}{30} 4^n + \mathcal{O}\left(2^{3n/2}\right) \quad \textit{if } n \textit{ is odd}.
\end{aligned}$$

This result improves on the C-NOT count for a similar decomposition for dense unitaries based on Householder reflections given in [14] which achieves 4^n to leading order. Note however that in the case of dense unitaries there are decompositions that achive better counts [4].

4.2 Sparse isometries

Now we consider the Householder decomposition for sparse isometries. Again the decomposition works for

any isometry, but now the number of C-NOT gates depends on the number of non-zero entries and their structure. The method yields lower C-NOT counts than the decomposition for dense isometries if the isometry is sufficiently sparse. We do not specify a precise number of zeros an isometry should have in order to be considered sparse, but use W to denote isometries in the context of methods designed to make use of sparseness.

4.2.1 Decomposition of sparse isometries

The basic idea of the sparse Householder decomposition is that if the columns of W are sparse, then so are the Householder vectors used in the decomposition. Therefore we can use our sparse state preparation method (Corollary 5) to save C-NOTs. The main obstacle is fill-in generated by the Householder reflections, i.e., where zero entries of the isometry are removed by the reflection. Corollary 11 implies that such fill in is relatively small. It can be further reduced by decomposing the columns of the isometry in a well-chosen order.

More precisely let ρ be a permutation of the rows of W and let σ be a permutation of its columns. We call the pair (ρ, σ) an elimination strategy. Then the sparse Householder decomposition proceeds as in the dense case, except that at step i we reduce column $\sigma(i)$ to row $\rho(i)$. If we implement the Householder reflections up to a permutation and a diagonal gate, then, after all reductions, the isometry will be a row permutation of a diagonal isometry. More precisely define $W_0 = W$ and iteratively

$$|w_i\rangle = W_i |\sigma(i)\rangle\,, \quad G_i = \Delta\Pi^i H_{w_i,\tilde{\rho}(i)}, \quad W_{i+1} = G_i W_i$$

for $i = 0, \ldots, 2^m - 1$ and where Π^i is a permutation gate and $\tilde{\rho}(i) = (\Pi^{i-1} \cdots \Pi^1 \Pi^0 \circ \rho)(i)$ with $\circ$ denoting the action of a permutation gate on a permutation. Then $G_{2^m-1} \cdots G_0 W = \Pi_n I_{n,m} \Delta_m$. The following two lemmas show how to decompose permuted diagonal gates and how to implement Householder reflections up to a permutation and a diagonal gate.

Lemma 15. *Let Π_n be a permutation gate on n qubits and Δ_m a diagonal gate on m qubits. An isometry of the form $\Pi_n I_{n,m} \Delta_m$ can be implemented using*

$$\mathcal{N}_{\Pi I \Delta}(n, m) \leq \mathcal{N}^{\Delta}_{\mathrm{Piv}}(n, m) + \mathcal{N}^{\Delta}_{\Pi}(m) + \mathcal{N}_{\Delta}(m)$$

C-NOTs. *Explicit counts are given in Table 2.*

Proof. Consider the vector $|u\rangle$ formed by summing the columns of $\Pi_n I_{n,m} \Delta_m$ and the pivoting algorithm (Algorithm 1) applied to this. If the gates required to pivot $|u\rangle$ are applied to $\Pi_n I_{n,m} \Delta_m$, all the 2^m non-zero entries of $\Pi_n I_{n,m} \Delta_m$ are moved to the top. [Note that when taking the counts from Remark 4, we replace $\mathrm{nnz}(v)$ by 2^m.] The resulting isometry has the form $I_{n,m} \Pi_m \Delta_m$. Since it leads to the same structure, it suffices to perform the pivoting up to a diagonal. It remains to implement a permutation on m qubits (up to diagonal) and a diagonal gate on m qubits. □

Lemma 16. *Let $|v\rangle$ be a state on n qubits and let $\mathrm{nnz}(v)$ denote the number of non-zero entries of $|v\rangle$ in the computational basis. Let $s = \lceil \log_2 \mathrm{nnz}(v) \rceil$. Then the standard Householder reflection H_v can be implemented up to diagonal and permutation gates using*

$$\mathcal{N}^{\Delta\Pi}_{H}(n, s) \leq 2\mathcal{N}_{\mathrm{SP}}(s) + \mathcal{N}^{\Delta}_{\mathrm{Piv}}(n, s) + \mathcal{N}_{H_0}(n)$$

C-NOTs. *Explicit counts are given in Table 2.*

Proof. It follows from Lemma 8 and Eq. (1) that H_v can be implemented up to diagonal and permutation gates as

$$(\Delta\,\mathrm{Piv}_v) H_v = (I_{n-s} \otimes \mathrm{SP}_{\tilde{v}}) H_0 (I_{n-s} \otimes \mathrm{SP}_{\tilde{v}})^\dagger \Delta\,\mathrm{Piv}_v,$$

which gives the claimed bound. □

Now we can give the C-NOT counts for the sparse Householder decomposition. First we define

$$\mathrm{elim}(W, \rho, \sigma) = \sum_{i=0}^{2^m-1} (\mathrm{nnz}(w_i) - 1) \tag{3}$$

to be the total number of eliminations during the Householder decomposition of W when using the elimination strategy (ρ, σ). Recall that $|w_i\rangle$ denotes the column reduced in the i^{th} step of the decomposition, which differs in general from $W|\sigma(i)\rangle$.

Lemma 17. *Let W be an isometry from m to n qubits and let (ρ, σ) be an elimination strategy. Then W can be implemented using*

$$\mathcal{N}_{\mathrm{iso}}(n, m) \leq \sum_{i=0}^{2^m-1} \mathcal{N}^{\Delta\Pi}_{H}(n, s(i)) + \mathcal{N}_{\Pi I \Delta}(n, m)$$

C-NOTs *where $s(i) = \lceil \log_2(1 + \mathrm{nnz}(w_i)) \rceil$. Explicit counts are given in Table 1.*

Proof. In the Householder decomposition the steps involve $H_{w_i,\tilde{\rho}(i)}$. This is a standard Householder reflection with respect to a vector that may have one additional non-zero element (see Eq. (2)). The bound given then follows from the sparse Householder decomposition. □

The count resulting from Lemma 17 depends on the chosen elimination strategy. The optimal strategy is the one that minimizes the amount of fill-in produced and therefore the number of eliminations required.

Remark 18. It can be beneficial to use the idea of the sparse Householder decomposition, without adhering to the exact form given above. For example, using a single standard Householder reflection we can implement a k-controlled single-qubit gate up to diagonal. This gate can be used to improve the C-NOT

count of the column-by-column decomposition [7]. Indeed, without loss of generality we can assume that the least significant qubit is the target qubit of the k-controlled single-qubit gate. Then the corresponding unitary is the identity matrix except for the 2×2 block in the bottom right corner. We can reduce the penultimate column (up to diagonal) using a standard Householder reflection with respect to $|1 \dots 10\rangle_n$ and two single-qubit gates for the state preparation and reverse state preparation. The C-NOT count is then that of a k-controlled NOT gate by a similar argument as in Lemma 9.

Remark 19. To obtain a more explicit bound we can plug in the counts from Table 2:

$$\begin{aligned}\mathcal{N}_{\text{iso}}(n,m) &\leq \sum_{i=0}^{2^m-1} [(n+16s(i)-5)(1+\text{nnz}(w_i))+16n] \\ &\quad + (n+34m-34)2^m \\ &\leq (17n-5)\sum_{i=0}^{2^m-1} \text{nnz}(w_i) + (34n+34m-39)2^m \\ &= (17n-5)\,\text{elim}(W,\rho,\sigma) + (51n+34m-44)2^m,\end{aligned}$$

where we have used $s(i) \leq n$. The given bound is valid with one dirty ancilla.

Remark 20. For very sparse isometries on a small number of qubits, the implementation cost of the H_0 gates used to implement standard Householder reflections (see Lemma 16) dominates the total number of C-NOTs required to implement the sparse isometry. However, on some experimental architectures it may be possible to directly implement H_0 gates by evolving the quantum system with an adapted Hamiltonian (see, e.g., [19]). Such architectures would hence allow for a very low cost implementation of sparse isometries on a small number of qubits.

4.2.2 Fill-in and envelopes

In order to gain as much advantage as possible from the sparseness of an isometry, we need to minimize fill-in as much as possible, which corresponds to choosing ρ and σ so as to minimize $\text{elim}(W,\rho,\sigma)$. Reducing a column of W in general affects other columns and creates new non-zero entries. Due to the orthogonality of the columns however, Householder reflections create little fill-in when applied to isometries. In fact, it follows immediately from Corollary 11 that when reducing column j to row i fill-in can only occur in columns that are non-zero in the i^{th} entry and fill-in is confined to the rows where $W|j\rangle$ is non-zero.

We use matrix envelopes to give a bound on the amount of fill-in the sparse Householder decomposition produces. The envelope of a sparse matrix gives for each column of the matrix the row index of the lowest non-zero element (i.e., the largest row index of a non-zero element) in that column or any previous column.

Definition 21. Let W be an isometry from m to n qubits. The *envelope* of W is defined to be the function $\text{env}_W : \{0,1,\dots,2^m-1\} \to \{0,1,\dots,2^n-1\}$ that maps each column index j to the smallest row index $\text{env}_W(j)$ such that $\langle i|W|j\rangle = 0$ for all $i > \text{env}_W(j)$ and such that $\text{env}_W(j+1) \geq \text{env}_W(j)$.

Definition 22. Let W be an isometry from m to n qubits. Define $\text{ed}(W) = \sum_{j=0}^{2^m-1}(\text{env}_W(j) - j)$. Then $\text{ed}(W)$ denotes the number of entries between the envelope and the diagonal of W.

The definition of $\text{ed}(W)$ is motivated by the following result.

Lemma 23. *Let W be a sparse isometry from m to n qubits and let (ρ,σ) be arbitrary row and column permutations. Let Π_ρ and Π_σ denote the corresponding permutation matrices. Then* $\text{elim}(W,\rho,\sigma) \leq \text{ed}(\Pi_\rho W \Pi_\sigma)$.

Proof. Let $\tilde{W} = \Pi_\rho W \Pi_\sigma$ for some permutations (ρ,σ). Reducing the columns of W according to (ρ,σ) is equivalent in terms of fill-in to reducing $\tilde{W}$ according to the trivial elimination strategy (ι,ι) where ι denotes the identity permutation, i.e., $\text{elim}(W,\rho,\sigma) = \text{elim}(\tilde{W},\iota,\iota)$. It follows from Corollary 11 that the fill-in for $\tilde{W}$ is confined to entries (i,j) with $i \leq \text{env}(j)$. Due to orthogonality of the columns, if we proceed in column order we never need to eliminate any elements above the diagonal, so $\text{elim}(\tilde{W},\iota,\iota) \leq \text{ed}(\tilde{W})$. □

Finding the row and column permutations that minimize the envelope is a computationally difficult task. Methods for similar problems have been considered in the context of sparse matrix decompositions [20]. Note that once a column permutation is fixed, the optimal row permutation can be found in the following straightforward way.

Let W be an isometry from m to n qubits. Consider the following algorithm for constructing a modified isometry W'.

Algorithm 2.

1. Set $i = 0$, $j = 0$ and $W' = W$.
2. Set k to be the number of non-zero elements in the column with index j with row indices greater than or equal to i.
3. If $k \neq 0$, permute the rows with indices greater than or equal to i such that these k non-zero elements have row indices i to $i+k-1$, assign the new isometry to W' and set $i = i+k$.
4. If $j < 2^m - 1$, set $j = j+1$ and return to Step 2, otherwise output W'.

Lemma 24. *Let W be an isometry from m to n qubits and W' be the output after applying Algorithm 2 to W. For all row permutations ρ' we have* $\text{env}_{W'}(j) \leq \text{env}_{\Pi_{\rho'}W}(j)$ *for all j.*

Proof. Given a column index j, let $t(j)$ be the number of rows such that for each of the columns with indices smaller than or equal to j, those rows have at least one non-zero element, i.e.,

$$t(j) := 2^n - |\{i : \langle i| W |j'\rangle = 0 \;\forall\; j' \in \{0, 1, \ldots, j\}\}| .$$

It follows that for any row permutation ρ' we have $\mathrm{env}_{\Pi_{\rho'} W}(j) \geq t(j) - 1$ for all j (recall that $\mathrm{env}_{\Pi_{\rho'} W}$ is a non-decreasing function by definition). However, by construction, $\mathrm{env}_{W'}(j) = t(j) - 1$ for all j, from which the claim follows. □

The difficult part is therefore finding the best column permutation. In this work we consider a simple greedy algorithm for finding a good column permutation.

Algorithm 3.

1. Set $i = 0, j = 0$ and $W' = W$.
2. Set M to be the submatrix of W' formed by only considering rows with row index greater than or equal to i and columns with column index greater than or equal to j.
3. Pick one of the columns of M with the fewest non-zero elements and set k to be the number of non-zero elements.
4. Permute the columns of W' with column index greater than or equal to j such that when restricting the permutation to M, the chosen column becomes the first column of M. Then permute the rows of W' with row index greater than or equal to i such that when restricting the permutation to M, all non-zero elements of the chosen column are moved to the top. Set $i = i + k$, $j = j + 1$.
5. If $j < 2^m$ and $i < 2^n$ return to Step 2, otherwise output W'.

This algorithm corresponds to minimizing the increment of the envelope at each step. More information on an efficient way to implement it can be found in Section 5.3.

4.3 Fixed envelope method

The asymptotic C-NOT count for the sparse Householder decomposition contains an undesirable factor n stemming from Lemma 2. We now show how this factor can be avoided if we give up some control on the amount of fill-in.

For this we make use of the decrement gate Dec_n which subtracts 1 in the computational basis (modulo 2^n), i.e., $\mathrm{Dec}_n = \sum_{i=0}^{2^n-1} |i\rangle\langle i \oplus 1|$ where $\oplus$ denotes addition modulo 2^n. This gate can be implemented using $\mathcal{O}(n)$ C-NOTs and one ancilla qubit [21]. The method is illustrated in Figure 2.

Lemma 25. *Let W be a sparse isometry from m to n qubits and let (ρ, σ) be some row and column permutations. Then W can be implemented using*

$$\mathcal{N}_{\mathrm{iso}}(n, m) \leq \mathcal{N}_{\Pi}^{\Delta}(n) + \sum_{i=0}^{2^m-1} \mathcal{N}_{\mathrm{col}}(s(i)) + \mathcal{N}_{\Pi}^{\Delta}(m) + \mathcal{N}_{\Delta}(m)$$

C-NOTs *where $s(i) = \lceil \log_2(1 + \mathrm{env}_{\Pi_\rho W \Pi_\sigma}(i) - i) \rceil$ and where*

$$\mathcal{N}_{\mathrm{col}}(s(i)) = 2\mathcal{N}_{\mathrm{SP}}(s(i)) + \mathcal{N}_{H_0}(n) + \mathcal{N}_{\mathrm{Dec}_n}.$$

Explicit counts are given in Table 1.

Proof. We consider reducing W in the following way. First apply the row permutation ρ up to diagonal. Then reduce the columns in the order given by the column permutation σ. For each column, use a Householder reflection to reduce it to the topmost row. Apply the decrement gate and then move to the next column. After all columns have been reduced in this way we apply $X^{\otimes n-m} \otimes I_m$. The resulting isometry has the form $I_n \Pi_m \Delta_m$ which can be reduced to the identity by applying a permutation on m qubits up to diagonal and a diagonal gate on m qubits.

By construction, before each Householder reflection, all non-zero entries in the column being reduced are in the topmost $2^{s(i)}$ positions. We can hence perform the Householder reflections using the method of Lemma 16 but omitting the pivoting steps. □

Remark 26. To obtain a more explicit bound we can plug in the counts from Table 1 to obtain

$$\mathcal{N}_{\mathrm{iso}}(n, m) \leq 4\,\mathrm{ed}(\Pi_\rho W \Pi_\sigma) + \mathcal{O}(n2^n).$$

This bound is valid with one dirty ancilla. The factor 4 stems from the fact that each Householder reflection uses state preparation twice and each state preparation acts on $s(i)$ qubits where $2^{s(i)}$ is at most twice the height of the envelope $1 + \mathrm{env}_{\Pi_\rho W \Pi_\sigma}(i) - i$ in column i of $\Pi_\rho W \Pi_\sigma$.

4.4 No fill-in method

Using a clean ancilla qubit we can avoid fill-in altogether. The method is illustrated in Figure 3.

Lemma 27. *Let W be a sparse isometry from m to n qubits. Then, using one additional clean ancilla, W can be implemented using*

$$\mathcal{N}_{\mathrm{iso}}(n, m) \leq \sum_{i=0}^{2^m-1} \mathcal{N}_{H}^{\Delta\Pi}(n+1, s(i)) + \mathcal{N}_{\Pi I \Delta}(n+1, m)$$

C-NOTs *where $s(i) = \lceil \log_2(1 + \mathrm{nnz}(W |i\rangle)) \rceil$. Explicit counts are given in Table 1.*

Proof. We implement the isometry $\tilde{W}$ from m to $n+1$ qubits defined by

$$\tilde{W} |i\rangle = |0\rangle \otimes W |i\rangle$$

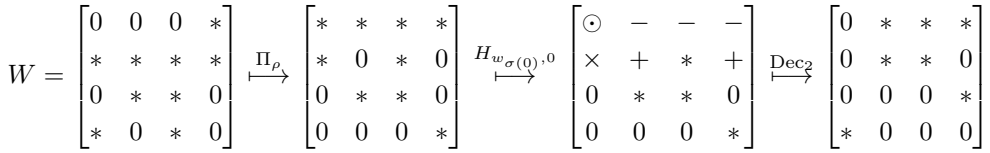

Figure 2: **Illustration of the first step of the fixed envelope method.** Here $*$ represents an arbitrary complex entry, $\odot$ denotes the target entry of the reduction, $\times$ stands for an entry that was eliminated, $-$ denotes entries eliminated due to the orthogonality constraint and $+$ means that fill-in occurs. Here $\sigma(0) = 0$.

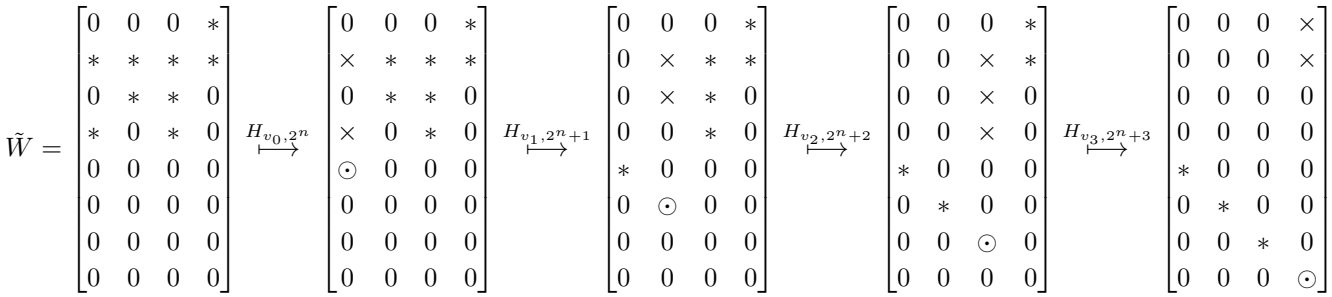

Figure 3: **Illustration of the no fill-in method.** The clean ancilla leads to the empty 4×4 block at the start. Here $*$ represents an arbitrary complex entry, $\times$ stands for an entry that was eliminated and $\odot$ denotes the target entry of each reduction. We actually implement each of the Householder reflections up to permutation, so the final state is a row permutation of that shown (for simplicity we did not depict this).

which in the computational basis is just W stacked on top of a zero matrix of the same size. Then each of the 2^m columns can be reduced to one of the 2^n zero rows without creating any fill-in. These reductions can be implemented using Householder reflections up to a diagonal and permutation gate as in Lemma 16. Then using Lemma 15 we reduce the resulting permuted diagonal isometry to the identity. □

5 Classical Complexity

In this section we compute the classical worst-case time complexity of some decompositions presented in this work. We compare the dense Householder decomposition to other known methods and we propose a sparse storage format which is well adapted to the sparse Householder decomposition.

5.1 Dense isometries

First we consider the dense case. For each column we have to compute the corresponding Householder vector (see Eq. (2)) and apply the Householder reflection to the entire isometry and produce the circuit implementing the Householder reflection. Computing the vector takes $\mathcal{O}(2^n)$ and applying the reflection takes $\mathcal{O}(2^{m+n})$. Producing the circuit takes $\mathcal{O}(2^{3n/2})$ according to Appendix B.4 of [10]. Thus the classical complexity is $\mathcal{O}(2^{m+n}(2^m + 2^{n/2}))$. For comparison, the column-by-column decomposition requires $\mathcal{O}(n2^{2m+n})$, and Knill's decomposition and the Cosine-Sine decomposition both require $\mathcal{O}(2^{3n})$ [10]. A high performance implementation for a decomposition of dense unitaries based on Householder reflections is presented in [14].

5.2 Sparse state preparation

The non-zero pattern of a sparse state on n qubits with at most 2^s non-zero entries can be compactly stored as a list of the n-bit indices of the non-zero entries. This requires $\mathcal{O}(n2^s)$ space.

The pivoting algorithm, Algorithm 1, can be implemented with some greedy optimizations. First there are $\binom{n}{s}$ possible splittings for the n qubits. We will think of the vector as being reshaped into a two dimensional array with 2^s rows and 2^{n-s} columns corresponding to the chosen qubit splitting. If the number of possible splittings is small enough, we try all splittings and choose the one with the largest number of non-zero elements in one column and this will be the target column. Otherwise one can randomly sample a fixed number of splittings and use the best one. Second, in each insertion step we choose an element not yet in the target column for which the cost of inserting it into the target column is minimal and perform the insertion. We iterate this until all elements are in the target column.

The insertion of one element into the target column can be implemented using one s-controlled NOT gate

and $d-1$ C-NOTs, where d is the Hamming distance between the index of the non-zero entry and the index of the target entry. Thus we want to find a non-zero entry outside the target column and a zero entry in the target column for which the Hamming distance is minimal. We do this by finding for each non-zero entry outside the target column the closest zero entry in the target column. The Hamming distance can be written as $d = d_c + d_r$ where d_c is the Hamming distance obtained when restricting the indices to the column indices, and d_r is the part corresponding to the row indices. For each non-zero entry, we can compute d_c in $\mathcal{O}(n-s)$, which adds up to $\mathcal{O}((n-s)2^s)$ for all non-zero entries outside the target column. We can compute d_r for all non-zero entries at the same time in $\mathcal{O}(s2^s)$ by using breadth first search on the s-dimensional hypercube with multiple starting vertices, given by the row indices of the zero entries in the target column. More precisely, we store a list of length 2^s, where each entry corresponds to one row and stores the minimal distance d_r to some free row of the target column. The entries corresponding to free rows are initialized with distance 0 and all other entries are initialized with distance ∞. Then we perform the usual breadth first search on the graph whose vertices are given by the entries of the list and whose edges connect any pair of entries whose indices have Hamming distance one. Performing the insertion takes $\mathcal{O}(n2^s)$. The entire circuit implementing sparse state preparation with the greedy optimizations mentioned above can thus be computed in time $\mathcal{O}(\binom{n}{s} + n2^{2s})$.

5.3 Sparse isometries

To store a sparse isometry W we store two arrays R and C of size 2^n and 2^m respectively. For a given row index i, $R(i)$ stores a reference to a balanced tree (e.g. a red-black tree) containing for each non-zero element of the i^{th} row a triplet of the form $(i, j, \langle i|W|j\rangle)$. The elements of the tree are sorted according to the key j. Analogously we define $C(j)$ to store a reference to a tree containing the non-zero elements of the j^{th} column. This requires $\mathcal{O}(2^n + 2^m + \text{nnz}(W))$ space. Given row and column indices i and j, the corresponding entry can be created, read, modified or deleted in time $\mathcal{O}(n)$.

We now show how to reduce column j to row i. From Corollary 11 we know that the modified entries are those in column j and row i and those with indices s and t such that $s \in C(j)$ and $t \in R(i)$. First we iterate over all choices for $s \neq i$ and $t \neq j$ and create or modify the entries according to Lemma 10. Then we set the entries in column j and row i to zero except for the entry with indices (i, j) which is determined by Lemma 10. Thus each reduction can be done in time $\mathcal{O}(n \,\text{mod})$ where mod denotes the number of modified elements.

In Algorithm 3 we presented a greedy method for constructing a permutation of an isometry leading to a small envelope. For a sparse isometry W given in the data structure described above, the corresponding row and column permutations ρ and σ can be computed iteratively as follows. Using the notation from Algorithm 3, we only store the submatrix M, which is initially set to W. In each step we find the sparsest column of M, which will be the next column in the column permutation, and the rows containing the non-zero elements of this column, which will be the next rows in the row permutation. Then we simply delete the non-zero elements in the chosen column and rows and iterate the procedure. In order to find the sparsest column in each iteration, we maintain a minheap storing for each column the number of non-zero elements. The smallest element can be removed in time $\mathcal{O}(n)$ which yields the sparsest column. Then we can delete the elements as described above, each in time $\mathcal{O}(n)$. For every deleted element we decrease the number of non-zero elements in the containing column by one, and thus we may have to reorder the minheap. But this can also be done in time $\mathcal{O}(n)$. The procedure stops when all non-zero elements have been deleted after total time $\mathcal{O}(n \,\text{nnz}(W))$.

6 Results for sparse state preparation in practice

We compare the C-NOT counts resulting from our sparse state preparation scheme presented in Section 2 to the dense case. The implementation of the sparse state preparation scheme is described in Section 5.2. In order to improve the classical computation time, we do not consider all possible qubit splittings, but randomly sample 100 splittings and choose the one with the largest number of non-zero elements in one column. We use the dense state preparation scheme from [5], which achieves near optimal C-NOT counts for arbitrary dense states. All of the methods are implemented in UNIVERSALQCOMPILER [10]. The results are presented in Figure 4. These indicate the advantage we gain by taking into account the sparseness. Note however, that the dense case outperforms the sparse case for fairly dense states (where the cost of pivoting is not compensated by the smaller state preparation). It is also worth noting that the counts found in practice are significantly smaller than our upper bounds.

Acknowledgements

The authors thank Vadym Kliuchnikov for pointing out Ref. [13].

RC is supported by EPSRC's Quantum Communications Hub (grant numbers EP/M013472/1 and EP/T001011/1). RI acknowledges support from the

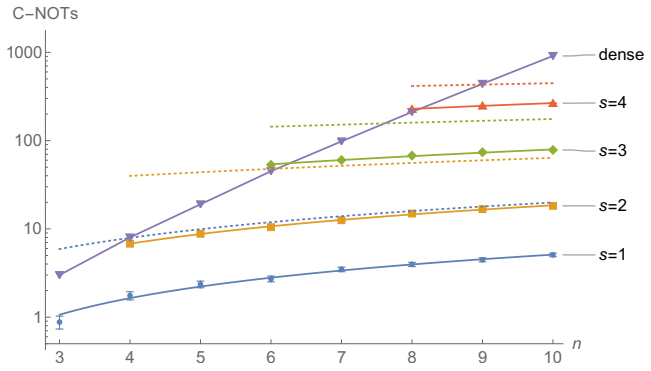

Figure 4: **Performance of sparse state preparation.** This shows the average number of C-NOT gates produced by our sparse state preparation scheme for sparse states on n qubits with 2^s non-zero entries, whose positions are chosen uniformly at random. Note that the actual values of the non-zero entries do not influence the counts. We used 200 trials and the error bars indicate twice the standard error of the mean (about 95% confidence). Note that for $s \geq 2$ the error bars are too small to be visible. We added linear best fit lines (except in the dense case). No ancillas were used for our implementation. The dashed lines show the upper bound of $(n + 6s - 7 + 23/24)2^s$ from Table 2, based on $\lceil \frac{s}{2} - 1 \rceil$ clean ancillas. Although our implementation does not use ancillas, it nevertheless beats this bound.

Swiss National Science Foundation through SNSF project No. 200020-165843 and through the National Centre of Competence in Research *Quantum Science and Technology* (QSIT).

References

[1] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," Phys. Rev. A **52**, 3457–3467 (1995).

[2] V. V. Shende, I. L. Markov, and S. S. Bullock, "Minimal universal two-qubit controlled-not-based circuits," Phys. Rev. A **69**, 062321 (2004).

[3] V. V. Shende, I. L. Markov, and S. S. Bullock, "Smaller two-qubit circuits for quantum communication and computation," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, Vol. 2 (2004) pp. 980–985.

[4] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum-logic circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**, 1000–1010 (2006).

[5] M. Plesch and Č. Brukner, "Quantum-state preparation with universal gate decompositions," Phys. Rev. A **83**, 032302 (2011).

[6] V. Bergholm, J. J. Vartiainen, M. Möttönen, and M. M. Salomaa, "Quantum circuits with uniformly controlled one-qubit gates," Phys. Rev. A **71**, 052330 (2005).

[7] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, "Quantum circuits for isometries," Phys. Rev. A **93**, 032318 (2016).

[8] E. Knill, "Approximation by quantum circuits," e-print arXiv:quant-ph/9508006 (1995).

[9] R. Iten, R. Colbeck, and M. Christandl, "Quantum circuits for quantum channels," Physical Review A **95**, 052316 (2016).

[10] R. Iten, O. Reardon-Smith, L. Mondada, E. Redmond, R. Singh Kohli, and R. Colbeck, "Introduction to UniversalQCompiler," e-print arXiv:1904.01072 (2019).

[11] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **22**, 710–722 (2003).

[12] S. P. Jordan and P. Wocjan, "Efficient quantum circuits for arbitrary sparse unitaries," Phys. Rev. A **80**, 062301 (2009).

[13] V. Kliuchnikov, "Synthesis of unitaries with Clifford+T circuits," e-print arXiv:1306.3200 (2013).

[14] T. G. de Brugière, M. Baboulin, B. Valiron, and C. Allouche, "Quantum circuits synthesis using Householder transformations," Computer Physics Communications **248**, 107001 (2020).

[15] D. Maslov, "Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization," Phys. Rev. A **93**, 022311 (2016).

[16] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," J. ACM **5**, 339–342 (1958).

[17] P. A. Ivanov, E. S. Kyoseva, and N. V. Vitanov, "Engineering of arbitrary U(n) transformations by quantum Householder reflections," Phys. Rev. A **74**, 022323 (2006).

[18] B. Torosov, E. Kyoseva, and N. Vitanov, "Fault-tolerant composite Householder reflection," Journal of Physics B: Atomic **48**, 135502 (2015).

[19] S. Fenner, "Implementing the fanout gate by a Hamiltonian," e-print arXiv:quant-ph/0309163 (2003).

[20] P. Heggernes and P. Matstoms, "Finding good column orderings for sparse QR factorization," in *Second SIAM Conference on Sparse Matrices* (1996).

[21] C. Gidney, "Factoring with $n + 2$ clean qubits and $n - 1$ dirty qubits," e-print arXiv:1706.07884 (2017).

A Details on Householder reflections

Given two unit vectors $|v\rangle$ and $|w\rangle$ we want to construct a map that sends $|v\rangle$ to $e^{i\theta}|w\rangle$ for some real θ.

This can be achieved using a Householder reflection with respect to the unit vector

$$|u_\theta\rangle = \frac{|v\rangle - e^{i\theta}|w\rangle}{\| |v\rangle - e^{i\theta}|w\rangle \|}.$$

Let us first compute the normalization

$$\begin{aligned}\| |v\rangle - e^{i\theta}|w\rangle \|^2 &= (\langle v| - e^{-i\theta}\langle w|)(|v\rangle - e^{i\theta}|w\rangle) \\ &= 2 - 2\,\mathrm{Re}(e^{i\theta}\langle v|w\rangle) \\ &= 2\,\mathrm{Re}(1 - e^{i\theta}\langle v|w\rangle) \\ &= 2\,\mathrm{Re}(z).\end{aligned}$$

where $z := 1 - e^{i\theta}\langle v|w\rangle$. If $z = 0$ no transformation is needed, so assume $z \neq 0$.

Consider the generalized Householder reflection $H^\phi_{u_\theta}$ acting on $|v\rangle$. We obtain

$$\begin{aligned}H^\phi_{u_\theta}|v\rangle &= |v\rangle + \frac{e^{i\phi}-1}{2\,\mathrm{Re}(z)}(|v\rangle - e^{i\theta}|w\rangle)(1 - e^{-i\theta}\langle w|v\rangle) \\ &= |v\rangle + \frac{e^{i\phi}-1}{2}\frac{\bar{z}}{\mathrm{Re}(z)}(|v\rangle - e^{i\theta}|w\rangle).\end{aligned}$$

Requiring $H^\phi_{u_\theta}|v\rangle \propto |w\rangle$ leads to the following condition

$$\frac{e^{i\phi}-1}{2}\frac{\bar{z}}{\mathrm{Re}(z)} = -1 \Leftrightarrow e^{i\phi} = 1 - \frac{\bar{z}+z}{\bar{z}} = \frac{-z}{\bar{z}}$$

which we can also write as $\phi = \pi + 2\arg(z) \mod 2\pi$.

Note that numerical instabilities arise when the norm of $|v\rangle - e^{i\theta}|w\rangle$ is very close to zero. In such cases we can either increase the precision of the computation, or note that for very small norms ignoring the rotation is equivalent to allowing a small error.

A.1 Standard Householder reflection

For a standard Householder reflection we have $\phi = \pi$. Now choose $\theta = \pi - \arg(\langle v|w\rangle)$ or $\theta = 0$ if $\langle v|w\rangle = 0$. This implies $z = 1 + |\langle v|w\rangle| \neq 0$. Then we define

$$H_{v,w} = H_{u_\theta}.$$

This map has the property that $H_{v,w}|v\rangle = e^{i\theta}|w\rangle$.

Note that in this case z is never close to 0. Since the decompositions presented in the main text only use standard Householder reflections, we do not have to deal with numerical instabilites there.

A.2 Generalized Householder reflection

If we want to get rid of the phase $e^{i\theta}$ we have to use a generalized Householder reflection. Setting $\theta = 0$ implies that $z = 1 - \langle v|w\rangle$ which might lead to numerical instabilities. Then $\phi = \pi + 2\arg(z)$. We define

$$\tilde{H}_{v,w} = H^\phi_{u_\theta},$$

so that $\tilde{H}_{v,w}|v\rangle = |w\rangle$.

B Proofs for the dense Householder decomposition

Here we give the proofs of Lemma 13 and Corollary 14 stated in the main text.

Proof of Lemma 13. This follows analogously to the proof of Theorem 1 of [7], so we only point out the necessary modifications. The idea is to decompose the isometry via

$$V = S_{2^m-1}H_0^{\phi_{2^m-1}}S^\dagger_{2^m-1}\ldots S_0 H_0^{\phi_0}S_0^\dagger, \qquad (4)$$

where S_i implements state preparation of the i^{th} column of V. Our modification is to replace the 2^m generalized Householder reflections by standard Householder reflections, and then to correct for the difference by using a diagonal gate on m qubits at the end, i.e.,

$$V = (\Delta_m \otimes I_{n-m})S_{2^m-1}H_0S^\dagger_{2^m-1}\ldots S_0H_0S_0^\dagger. \qquad (5)$$

Lemma 9 shows that using one dirty ancilla qubit the standard Householder reflection H_0 can be implemented using $16n - 24$ C-NOTs instead of $16n^2 - 60n + 42$ used for the generalized version H_0^ϕ. The cost of the diagonal gate is $2^m - 2$ (see Table 3). □

Proof of Corollary 14. First we claim that a controlled isometry from $m-1$ to m qubits with $n-m$ controls can be implemented using at most $N(m-1,m) + 16(n-m)2^{m-1} + 2^{n-1} - 2^{m-1}$ C-NOTs where $N(m,n)$ denotes the upper bound for implementing an isometry from m to n qubits given in Lemma 13[1]. This follows from Lemma 13 and the fact that when implementing the controlled isometry by controlling all the gates in (5), we do not need to control the state preparation gates or their inverses (i.e., the S_i and $S_i^\dagger$ gates do not need controls). Thus the control does not affect the first three terms in the counts from Lemma 13.

To decompose the unitary, we start by reducing the first half of the columns using the inverse of a circuit implementing an isometry from $n-1$ to n qubits. This yields $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U_1$, where U_1 is an $n-1$ qubit unitary. We can then control on the first qubit and do the inverse of an isometry from $n-2$ to $n-1$ qubits and so on. At each step we reduce half of the remaining columns and requires the inverse of an isometry from $n-k-1$ to $n-k$ qubits with k controls, where $k = 0, 1, \ldots, n-1$.

The C-NOT count is thus

$$\sum_{k=0}^{n-1} N(n-k-1, n-k) + k2^{n-k+3} + 2^{n-1}(1-2^{-k}).$$

[1] We write N rather that $\mathcal{N}_{\text{iso}}$ to emphasise that the modification is only valid based on the technique of Lemma 13.

This can be used with the counts from Lemma 13 to upper bound the number of C-NOTs. To generate a clean bound on the leading order term, note that if n is even, the even values of k contribute $\frac{23}{48}\sum_{t=0}^{(n-2)/2} 2^{-4t}$ to the coefficient of 4^n, while the odd values of k contribute $\frac{115}{768}\sum_{t=0}^{(n-2)/2} 2^{-4t}$, giving a leading order of $\frac{161}{240}4^n$. Similarly, if n is odd, the even values of k contribute $\frac{115}{192}\sum_{t=0}^{(n-2)/2} 2^{-4t}$ to the coefficient of 4^n, while the odd values of k contribute $\frac{23}{192}\sum_{t=0}^{(n-2)/2} 2^{-4t}$, giving a leading order of $\frac{23}{30}4^n$. □

C Multi-controlled NOT gates

We denote a k-controlled NOT gate by $C_k(X)$. Using a single dirty ancilla qubit such gates can be decomposed with a linear number of C-NOT gates. We start by recalling two lemmas from [1, 7].

Lemma 28. *([1, Lemma 7.3]) Let $n \geq 5$ denote the total number of qubits. A $C_{n-2}(X)$ gate can be decomposed into two $C_k(X)$ and two $C_{n-k-1}(X)$, where $k \in \{2, 3, \ldots, n-3\}$.*

Lemma 29. *([7, Lemma 8]) Let $n \geq 5$ denote the total number of qubits and $k \in \{1, \ldots, \lceil \frac{n}{2} \rceil\}$, then we can implement a $C_k(X)$ gate with at most 8k-6 C-NOTs.*

Note that if $k \geq 5$ this bound can be reduced to $8k - 12$ [15]. However, we do not use this here for the convenience of having a single bound for all $k \leq \lceil \frac{n}{2} \rceil$. The desired decomposition of multi-controlled NOT gates using a single ancilla qubit follows.

Corollary 30. *Let $n \geq 3$ denote the total number of qubits. Then we can implement a $C_{n-2}(X)$ gate with at most $16n - 40$ C-NOTs.*

Proof. The case $n = 3$ is trivial and $n = 4$ can be done with 6 C-NOTs [1, Sec. VI A]. For $n \geq 5$ we can use Lemma 28, choosing $k = \lfloor \frac{n}{2} \rfloor$ (note that $2 \leq k \leq n-3$). We then have both $k \leq \lceil \frac{n}{2} \rceil$ and $n-k-1 \leq \lceil \frac{n}{2} \rceil$, so can use Lemma 29. This gives a count of at most

$$2(8k - 6) + 2(8(n - k - 1) - 6) = 16n - 40\,,$$

as required. □

D Permutation gates

One key feature of several of our methods is the use of permutations to adjust the form of the given isometry. Here we discuss the number of C-NOTs needed for these.

Lemma 31. *A permutation gate on $n \geq 3$ qubits can be performed without ancilla using $(27n-62)2^n + 44n^2 - 96n - 23$ C-NOTs.*

Proof. Permuting the rows of a state on n qubits corresponds to constructing a $2^n \times 2^n$ permutation matrix (i.e., a unitary matrix with one 1 in every row and column). Each such matrix corresponds to a permutation on 2^n objects. It is known that all permutations can be decomposed as a sequence of swaps. A permutation is *even* if it can be decomposed into an even number of swaps and otherwise it is *odd*. It is known that all even permutations on $n \geq 3$ qubits can be performed with at most n NOTs, n^2 C-NOTs and $3(2^n + n + 1)(3n - 7)$ Toffoli gates [11, Theorem 33]. A Toffoli gate can be performed up to a diagonal gate using 3 C-NOTs [1] and diagonal gates can be commuted with NOT, C-NOT and Toffoli gates up to another diagonal. Thus, ignoring the single-qubit (NOT) gates, any even permutation can be decomposed into at most $(27n-63)2^n+28n^2-36n-63$ C-NOTs without ancilla, up to a diagonal gate.

If we have an odd permutation on n qubits, we can apply an $(n-1)$-controlled NOT to make it an even permutation. Without an ancilla, we can do this as a $n-1$ controlled single-qubit unitary with an overhead of $16n^2 - 60n + 42$ C-NOTs (see Table 3). This leads to an overall C-NOT count of $(27n - 63)2^n + 44n^2 - 96n - 21$ C-NOTs without ancilla[2].

A diagonal gate on n qubits can be performed using $2^n - 2$ C-NOTs (see Table 3), leading to an overall C-NOT count of $(27n - 62)2^n + 44n^2 - 96n - 23$. □

Slightly lower counts are possible with ancillas, but these do not change the leading order so we do not consider them here for simplicity. Note also that the above bound is always less than $27n2^n$ for $n \geq 3$, so we use the latter as a simplification.

Any unitary on $n \geq 2$ qubits can be decomposed using at most $\lceil \frac{23}{48}4^n - \frac{3}{2}2^n + \frac{4}{3} \rceil$ C-NOTs [4] (without ancilla) which gives a better count for an arbitrary permutation for $n \leq 8$.

[In [11], the authors note that if we add a qubit on which we do not act, an odd permutation becomes even and hence any permutation on $n \geq 3$ qubits can be done using one ancilla and an even permutation on $n + 1$ qubits [11, Corollary 13]. However, this is significantly worse than the above bound.]

Lemma 32. *A permutation gate on $n \geq 2$ qubits can be performed with one dirty ancilla and at most $(18n - 26)(2^n - 1)$ C-NOTs.*

Proof. The idea is to use Householder reflections. We can write the permutation gate as $\sum_{i=0}^{2^n-1} |j(i)\rangle\langle i|$, where $|j(i)\rangle$ is a computational basis state. The Householder reflection $H_{j(i),i}$ takes $|j(i)\rangle \mapsto |i\rangle$ and $|i\rangle \mapsto |j(i)\rangle$ without affecting any other columns (cf. Corollary 11).

[2] Slightly lower counts are possible with ancilla but these do not change the leading order so we do not consider them here for simplicity.

Since $H_{j(i),i} = H_u$, where $|u\rangle = \frac{1}{\sqrt{2}}(|i\rangle - |j(i)\rangle)$, we can implement each Householder reflection along the lines given in Lemma 8. The state $|u\rangle$ can be reduced to $|i\rangle$ as follows. Let $|i\rangle = |i_1 i_2 \dots i_n\rangle$ and $|j(i)\rangle = |j_1 j_2 \dots j_n\rangle$ in the computational basis. Find an index k such that $j_k \neq i_k$. Controlling on the k^{th} qubit we can apply at most $n-1$ C-NOTs such that $|j(i)\rangle \mapsto |i_1 \dots i_{k-1} j_k i_{k+1} \dots i_n\rangle$ and $|i\rangle$ is unchanged. When applied to $|u\rangle$, this results in $|u'\rangle = \frac{1}{\sqrt{2}}(|i\rangle - X_k|i\rangle)$, where X_k is a NOT on the k^{th} qubit. Applying a single qubit rotation to the k^{th} qubit then maps $|u'\rangle$ to $|i\rangle$. Let us denote the reverse of these steps by $\mathrm{SP}_{i,u}$, so that $\mathrm{SP}_{i,u}|i\rangle = |u\rangle$. We have

$$H_{|u\rangle} = \mathrm{SP}_{i,u} H_i \mathrm{SP}_{i,u}^\dagger ,$$

where $H_i = I - 2|i\rangle\langle i|$ is either a Z or $-Z$ gate with $n-1$ controls, and hence has the C-NOT count of $C_{n-1}(X)$, which is $16n - 24$ if one dirty ancilla is available (see Table 3).

It follows that we can do $H_{j(i),i}$ using $18n - 26$ C-NOTs, and hence the whole permutation matrix using at most $(18n-26)(2^n - 1)$ C-NOTs. □

Again, for simplicity, we can bound this by $(18n - 26)2^n$.

E Knill-Householder decomposition

In this appendix we consider a generalized decomposition scheme that contains Knill's decomposition [8] and the Householder decomposition as special cases. Suppose U is a unitary that we want to implement and B is another unitary representing a change of basis. Assume that B and $\tilde{U} = B^\dagger U B$ are sparse. Let $|b_i\rangle = B|i\rangle$, then $|\tilde{u}_i\rangle := UB|i\rangle = U|b_i\rangle$ and

$$U = \sum_i U|b_i\rangle\langle b_i| = \sum_i |\tilde{u}_i\rangle\langle b_i| .$$

Consider the generalized Householder reflection $\tilde{H}_{\tilde{u}_0,b_0}$ mapping $|\tilde{u}_0\rangle$ to $|b_0\rangle$ (in this section we want this map to be exact and not up to a phase). In the basis formed by the columns of B this reduces the first column of $\tilde{U}$. Consider the effect on the second column $|\tilde{u}_1\rangle$. Since it is orthogonal to $|\tilde{u}_0\rangle$, this column will suffer fill-in if and only if it is not orthogonal to $|b_0\rangle$. In this case, fill-in will be confined to the subspace spanned by $|\tilde{u}_0\rangle$ and $|b_0\rangle$. This means that fill-in is determined by the structure of $\tilde{U}$ and works in the same way as for the Householder reflection in the standard basis. It is important to note however that state preparation is only efficient if $|\tilde{u}_0\rangle$ is sparse, i.e., if both $\tilde{U}$ and B are sufficiently sparse. Continuing in this fashion allows us to reduce U to the identity.

Remark 33. If we define B to be the unitary whose columns form an eigenbasis of U, then this scheme reduces to Knill's decomposition, and if B is the identity, it is essentially the Householder decomposition. Note however that in the case of the Householder decomposition we used standard Householder reflections and it was sufficient to implement them up to diagonal and permutation gates.

This method can be generalized to isometries by extending a given isometry V to a unitary U. This can be done such that U has at least $2^n - 2^m$ eigenvalues equal to 1 (see [8], [10, Lemma 5]). Let B be an n qubit unitary whose first $2^n - 2^m$ columns are eigenvectors of U with eigenvalue 1. Then $\tilde{U} = B^\dagger U B$ is blockdiagonal with a $(2^n - 2^m) \times (2^n - 2^m)$ trivial block and an $2^m \times 2^m$ non-trivial block. This observation can be used to reduce U as described above.